HLA Data Distribution Management: Design Document Version 0.5

Forward

This document represents the results of almost a year's effort by a team of individuals from several organizations, all focussed on defining the initial description of Data Distribution Management as it will be implemented in multiple RTIs. The baseline DDM definition period has been characterized by in-depth and substantive debate about the goals of DDM, and the most effective balance between generality and performance. The DDM team was formed at DMSO's request and guided through it's initial stages by Steve Seidensticker. The team's members include Steve Seidensticker, Daniel Van Hook and James Calvin and their team at MIT-Lincoln Laboratories, Jeffrey Steinman at Metron, Richard Fujimoto of Georgia Tech, Reed Little from CMU's Software Engineering Institute, Larry Mellon and Darrin West from SAIC, and myself. The results of the team's efforts can be seen in the HLA Interface Specification, for which Reed Little has responsibility, the initial DDM implementation developed by the team from MIT-Lincoln Laboratories, and this Design Document for which I have assumed responsibility from Jeff Steinman. This document will evolve and companion documents will be developed as I guide the DDM team through subsequent phases of DDM development and experimentation.

Katherine L. Morse
Senior Computer Scientist
Science Applications International Corporation

1. Introduction

The purpose of this technical report is to document the functionality of the High Level Architecture (HLA) data distribution management (DDM) services. This report is written primarily for those who are interested in how the DDM services work. While we have tried to explain how the data distribution management services can be used, this document is really not intended to function as a user's guide. Rather, it is a design document intended to provide the technical background for the data distribution services in HLA. Subsequent to early experiments with the current RTI, a use case study with lessons learned will be produced which will serve as a user's guide. It will contain specific scenarios, code examples, and metrics from specific experiments. It is anticipated that these two documents and any accompanying documents identified as necessary later will be "living" documents that evolve as new DDM services are considered in HLA.

The design and use approaches described in this document rely solely on the DDM services specified in the HLA Interface Specification. So, everything presented here is equally relevant to any RTI implementation. Details of specific RTI implementations are provided at the end.

Currently this document addresses real-time data distribution management. DDM in logical-time federates is still under investigation.

Section 2 briefly describes the HLA goals of data distribution management. Section 3 documents the multidimensional routing space approach. Section 4 then shows how to use the data distribution management services in a physically correct manner, taking into account (1) the fact that subscription regions and update regions are sampled discretely over time (not continuously), and (2) the effects of latency. Section 5 provides detailed design information for specific Run-Time Infrastructure (RTI) implementations. Section 6 is a short bibliography of useful documents that provide background to the filtering topic described in this report. Section 7 provides a list of terms and definitions that are relevant to the HLA DDM services.

2. Goals

The goal of HLA DDM services is to limit the messages received by federates in large distributed federations to those messages of interest in order to reduce (1) the data set required to be processed by the receiving federate and (2) the message traffic over the network. This functionality should be provided in a manner that is straightforward and easy to use. Thus, HLA data distribution management services are concerned with three things:

- **Efficiency:** Data distribution management services should involve minimum overheads in terms of *computations, message latencies*, and *memory usage* for all of its services. Expensive operations, such as string comparisons, complex or costly computations, excessive handshaking, etc., should be avoided whenever possible. A service must justify the overhead cost it incurs. An expensive service may be acceptable if it saves more than it costs.

- **Scalability:** Data distribution management services provided by the RTI should scale in terms of (1) *computational complexity* for handling requests, (2) *message traffic* and/or bandwidth for distributing information, and (3) *memory requirements* for storing attribute information, maintaining tables, etc. It is understood that the services are not required to scale better than the physical problem[1]. The parameters that normally affect scalability are: (a) the *number of federates* (or hosts) in the federation, (b) the *number of simulated entities* per federate, (c) the average complexity concerning the *interests of each entity* (i.e., an entity may have a number of different kinds of sensors), (d) the *interaction rates between federates* after one discovers relevant objects in the other, (e) the *locality of objects*[2], and indirectly (f) the *scenario.*

- **Interfaces:** Data distribution management services must support the *right interfaces* to provide the *right filtering functionality* that is needed in HLA federations. The interfaces should be provided in an easy-to-use manner. It is important to capture the right functionality in the HLA data distribution management services and to know where to draw the line between the RTI and software provided by the federates, which could be used to further automate much of the filtering for a federate. It is recognized that interfaces which describe limited functionality are usually easier to use than interfaces that describe complex and powerful functionality. The goal is to make the interfaces as easy to use as possible while supporting the essential HLA data distribution management services. The interfaces must also be sufficiently general that different underlying implementations may support the same common interfaces.

3. Data Distribution Management in HLA

Data distribution management in HLA includes all the mechanisms needed to discover objects and to enable and control the flow of their class attribute values and interaction parameters among federates via the RTI.

The publish and subscribe declaration management (DM) services provided by the RTI allow the federates to initiate the movement of this data. A federate which subscribes to an

---

[1] For example, if all of the simulated entities move to the same area of the battlefield and can therefore detect one another, then DDM cannot be expected to eliminate the n^2 nature of the physical problem.

[2] Objects that are local (i.e., within the same federate) can interact without requiring messages to be sent through the network. In large applications, it is sometimes possible to decide which federates create which objects. It often makes sense for objects that stay within the same physical region during the course of the simulation execution to be created within the same federate to provide better locality for the federation.

object class' attribute values (or to an interaction class) will receive all updates of the specified attribute values for all objects of that class which currently exist in the entire federation. This type of filtering has the benefit of eliminating delivery of attributes for whole classes of objects which are of no interest to the subscribing federate. We'll refer to this as *class-based* filtering. For a small federation or one with few objects created in each class, class-based filtering may be sufficient to support performance and scalability requirements. However, for many large federations with large numbers of objects in their classes more detailed filtering may be required to improve performance and scalability. The DDM services provide *value-based* filtering.

The DDM services allow a federate to receive the subscribed attributes selectively based on values or characteristics of the publishing federate. For example, using only DM services, a federate could subscribe to attributes of all fixed wing aircraft. However, if the "playbox" is large, not all fixed wing aircraft may be in sensor range of the subscribing federate. In this case, the federate could invoke DDM services to limit the aircraft whose attributes it receives to those within its sensor range.

3.1 A Conceptual Overview of Routing Spaces

To support efficient data distribution across a federation, the RTI provides a set of services which facilitate the explicit management of data distribution. The fundamental concept to support data distribution is called *routing spaces.* A routing space is a multidimensional coordinate system in which federates express an interest for either receiving data or sending data. The multidimensional routing space captures two important kinds of information:

- *Update Region*: A set of routing-space coordinate values of an object define a region[3] in the corresponding multidimensional routing space. As these values change dynamically over time, one can conceptualize a trajectory in a multidimensional space representing an object's volume in the routing space as it evolves over time.

- *Subscription Region*: The relevant discovery criteria, defined as bounded regions in the multidimensional routing space that describes the interests of subscribing federates[4]. Federates express their discovery interests as regions in this space which potentially change dynamically over time. It is possible for these regions to change in size as well as in position.

An object is discovered by a federate when the object's update region overlaps the federate's subscription region.

To use routing spaces, each federation defines the allowable routing spaces for the federation execution, including the dimensions (coordinates) of the routing space. The routing space dimensions are coordinated manually at federation planning time. Routing spaces are specified in the Federation Execution Data (FED) file with a name, the number of dimensions, and additional parameters.

Using declaration management services, federates specify by object class and attribute name (*Publish Object Class, Subscribe Object Class Attribute*) or by interaction class

---

[3] Update regions can be represented as n-dimensional rectangular volumes instead of points in a routing space. This can be useful for large objects, such as weather clouds, whose extents may overlap large routing space regions. Extended update regions can also be used to bound an object's true value in the routing space.

[4] Regions in a multidimensional routing space do not necessarily map to physical geographical regions. A region in a routing space should be thought of as an abstract volume with possibly more or possibly fewer than three dimensions, e.g. radio channels.

(*Publish Interaction Class, Subscribe Interaction Class*), the types of data they will send or receive. Routing spaces are used by federates to specify the distribution conditions for the specific data they are sending or expect to receive. Each federate decides which of the federation routing spaces are useful to them and defines the portions of those routing spaces that specify (from the federate's perspective) *regions,* or logical areas of interest particular to the federate, by putting bounds (*extents*) on the dimensions of the selected routing space. The federate then uses these regions to both specify conditions (*Create Subscription Region*) under which they expect to receive the object state data and interactions they specified using declaration management services and to specify conditions under which they are providing data *(Create Update Region* and *Associate Update Region).*

Specifying a subscription region, the federate tells the RTI to only deliver data which fall within the bounds (extents) of the region specified by that federate. Specifying an update region and associating that update region with a particular object instance, means that the federate will ensure that the characteristics of the object instance or interaction which map to the dimensions of the routing space fall within the bounds (extents) of the associated region at the time that the attribute update or send interaction call is issued. This implies that the federate is monitoring these added characteristics for each of the attributes owned by the federate. As the state of the objects change, the federate may need to either adjust the bounds on the associated regions (*Modify Region*) or change the association to another region (*Associate Update Region*).

The routing space, regions, and association data are used by the RTI to distribute data. When an update region and subscription regions of different federates overlap, the RTI ensures that attribute updates and interactions associated with the update region are routed to federates with subscription regions which overlap the sender's update region. The subscribing federates each receive only the object class attributes and interactions to which they subscribed, although they may receive individual updates outside their subscription region depending on the precision of the routing space implementation.

It is important to note that the dimensions of routing spaces need not necessarily map to attributes in the FED. The data distribution management services provide a federation the capability to specify data distribution on characteristics of objects other than those exchanged as part of federation execution. By specifying routing spaces and regions using attributes specified in the FED, the data distribution management services provide a mechanism to control data distribution based on values of attributes.

Each federate can create multiple update and subscription regions. Update regions are associated with individual objects that have been registered with the RTI via the object management service *Register Object*. A federate might have a subscription region for each sensor system being simulated.

Figure 1 shows one update region (U1) and two subscription regions (S1, S2) within a two dimensional routing space. In this example, U1 and S1 overlap and therefore attributes and interactions associated with U1 will be routed by the RTI to the federate that created S1. In contrast U1 and S2 do not overlap and attributes and interactions will not be routed from the federate that created U1 to the federate that created S2.

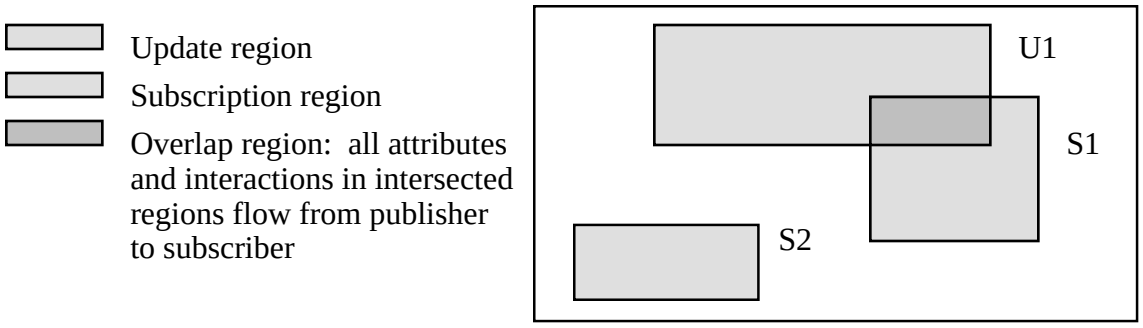

Figure 1: Two-dimensional Routing Space Example

3.2 Services Associated with Data Distribution Management

DDM services interoperate with HLA Object Management (OM) services and Declaration Management (DM) services to provide value-based filtering. OM services consist of the group of RTI services which deal with the creation, modification, and deletion of objects and the interactions they produce. DM establishes the flow of data between federates using the object class and attribute publication and subscription services. So, OM services register the objects of interest and reflect their attributes. DM services ensure that DDM services are aware of federates which are interested in the objects and attributes. And DDM services ensure that the subscribing federates only receive the attributes of objects which are of interest. Specifically, OM, DM, and DDM services perform the following functions:

- Create subscription and update regions (DDM)
- Associate an update region with an object and its published attributes (DDM)
- Subscribe to object, or interaction, classes (DM)
- Register or discover an object (OM)
- Remove a discovered object (OM)
- Update and reflect object attributes (OM)
- Send and receive interactions (OM)
- Modify or delete already existing routing space regions (DDM)

3.3 Object Discovery and Attribute Reflection

An example will serve to illustrate the specific flow and interaction of DDM, OM, and DM services which allow a federate to discover objects of interest and to receive the objects' attribute values as they are updated. Figure 2 shows an example of routing space usage based on position and range.

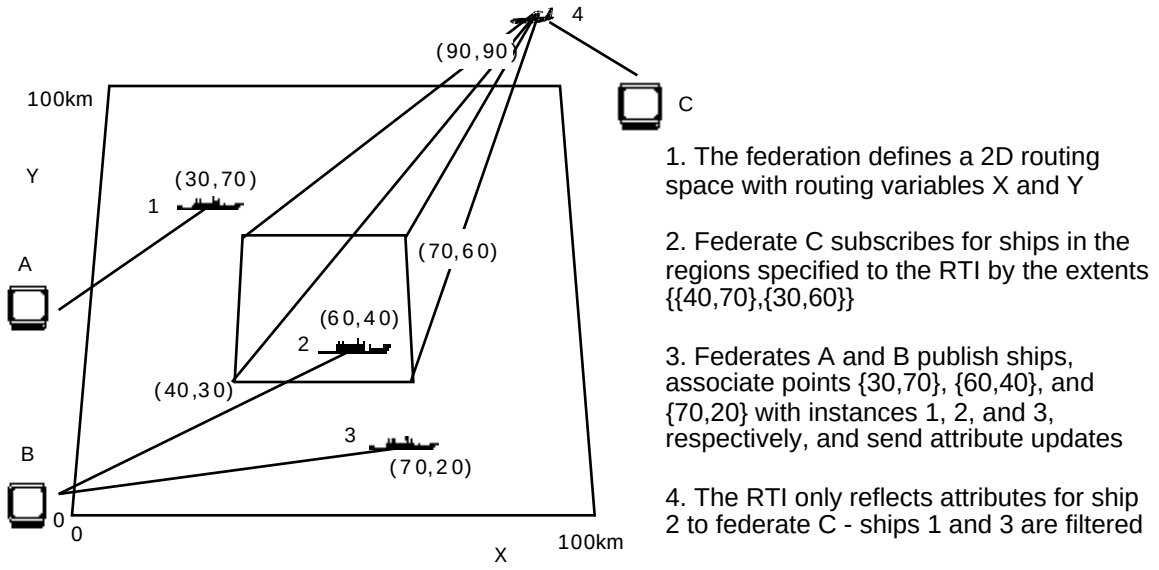

Figure 2: Playbox routing space example.

The federation begins the process by defining a routing space in the FED and creating the routing space with filter variables. Several important quantities are specified in each routing space, including:

- Name of the routing space
- Number of dimensions in the routing space

The federates must agree on the use of each dimension, i.e. the routing variables on each axis.

Objects are created by the federates and the objects are registered with the RTI using *Register Object*. As a result, IDs are associated with the objects. In the example, federate A's ship has ID 1; federate B's ships have IDs 2 and 3; federate C's aircraft has ID 4.

Object subscriptions can be defined for each existing object[5]. When an object that is participating in any of the routing spaces is created by a federate, two filter-related quantities need to be defined: (1) its initial update region (or regions) in each of its relevant routing spaces (*Create Update Region)*, so that other federates can discover it, and (2) its subscription regions in the routing spaces (*Create Subscription Region*), so that it can discover others. As time progresses, these two pieces of information can be updated using the *Modify Region* service according to a consistent and correct strategy that is associated with the routing space. It is up to the federates to define how to use a routing space correctly. In the example, federates A and B would create update regions around the initial positions of their ships. Federate C creates a subscription region corresponding to the sensor range of the aircraft.

When an object is created, its relevant published attributes need to be *associated* with a update regions if filtering is used for this object class. *Associate Update Region* ties together (1) a routing space, (2) the initial update region describing the object in the routing space, and (3) the corresponding set (or subset) of published attributes when the object is first created. Federates A and B associate the object IDs and published attributes of ships 1, 2, and 3 with their respective update regions.

---

[5] Subscription regions do not necessarily have to be associated with an object. For example, a plan view display federate may subscribe to a large portion of the virtual environment.

The RTI determines which federates should discover which objects by matching subscription and update regions. Federates are notified by the *Discover Object* service of objects that meet the federate's subscription requests. Object discovery is provided only once by the RTI to a federate even when multiple locally-defined subscription regions overlap a remote object's update region. Federate C is notified by the RTI to discover federate B's object 2.

On the publisher's side, object attributes are initially updated after the object has been created. Afterwards, a federate may update the attributes of the object as needed. The attributes are reflected in each of the federates that have discovered the object later in simulation time or wall clock time due to communication latencies via the *Reflect Attribute Values* service. Even though federates A and B may update the attributes of ships 1, 2, and 3, federate C only receives the updated attributes of ship 2 because it is the only one that federate C has "discovered". The updated attributes of ships 1 and 3 may be sent to other federates (not shown) that have discovered them via other subscription regions, or their attributes may not be sent anywhere if no other federate has subscribed to them.

It is possible for federates to try to predict when attributes will change and then update them ahead of time. However, it is much more straightforward, especially when integrating legacy simulations, to simply update the attributes as they change without trying to predict anything and then live with the fact that the receiving federates will reflect those values later in time due to network latencies. The data distribution management services work correctly in either case.

As an object's state variables change over time, it might become necessary to modify update or subscription regions. This normally occurs when routing space thresholds[6] are exceeded. Note that this does not have to occur every time an attribute is updated. When an update region or a subscription region is modified, the changes are passed to the RTI using the *Modify Region* service. The RTI then reassesses the matches between the modified region and all regions of the complementary type.

When a federate uses the *Delete Object* service, the RTI automatically removes its associated update regions from its internal structures and notifies federates reflecting the object's attributes to remove it. It is possible for a federate to have an object stop publishing or subscribing by making calls to *Delete Region*. These can be made for each region associated with an object so that the RTI knows not to continue sending or receiving information from other federates concerning a particular object.

When a federate's subscription region no longer contains the update region of an object, the data distribution management services must coordinate actions so that the receiving federate knows to delete its local representation of the object (assuming that no other locally defined subscription regions in any of the routing spaces contain the update region of the object). The RTI tells the subscribing federate to remove the object's representation[7] via the *Remove Object* service. *Remove Object* may also be triggered by the publishing federate invoking *Delete Object.*

The send/receive interaction services are similar to the create/discover object services. An example of an interaction class might be a detonation event that nearby objects should detect if they are close enough. Federates could subscribe to detonation interaction classes based on a routing space usage that supports entity positions and range-based proximity,

---

[6] Shortly after creating a region, a federate will receive a *changeThresholds* call from the RTI indicating how much a region's extents can change before the federate must inform the RTI of the new values.

[7] The HLA does not specify how federates internally represent remote objects which are discovered via the RTI. The HLA only specifies that federates receive *Discover Object* and *Remove Object* messages and behave consistently internally.

much the way they would subscribe to other proximity detectable entities. If a detonation interaction is sent through the RTI, then those federates with objects close enough to the interaction's detonation point will receive the interaction.

One difference between objects and interactions is that instead of providing attribute value updates, the interactions themselves are bundled with a fixed set of well-defined parameters in the send interaction service. When it comes to the discovery mechanism, data distribution management services equally apply to interactions and objects alike. However, interactions do not persist. This means that there is nothing comparable to the *Update Attributes, Delete Object,* or *Provide Attribute Value Update* services for interactions. It is implicitly assumed that the interaction will be discarded by each receiving federate after it has been processed.

4. Physically Correct Filter Strategies

Network latencies and lookahead restrict how tightly in time one federate can interact with another federate, for example, to request object attributes. In addition to network overheads, there are other factors that affect filtering. Some routing space coordinates used to describe update regions (motion, for example) change in a continuous manner over time. These are called *continuous coordinates*. Filtering strategies must somehow sample these continuous coordinates at rates that are not too high. Otherwise, the filter strategy might require too many computations and changes in the filter specifications. However, if the routing space coordinates are not sampled often enough, the filter will not be as responsive or as efficient as it otherwise could be.

Similarly, subscription regions can also change continuously over time. An efficient filtering strategy must not sample subscription regions too often or too infrequently for the same reasons. The filter strategy used by a federation must take into account the fact that true update regions and true subscription regions may not be accurately represented by the sampled regions that are represented as static regions at discrete points in time. Two ways of getting around this problem are (1) to widen subscription regions, and/or (2) to widen update regions in a physically correct manner. This must be done in a way that preserves true overlaps between the update and subscription regions.

There are other kinds of routing space coordinates that change their values in unpredictable manners. These are called *discrete coordinates*. Here, the routing space coordinate values remain constant unless they are changed to new values at discrete points in time.

4.1 Filtering on Discrete Coordinates

Discrete coordinates can be thought of as step functions[8]. Their true physical value remains constant until changed. Discrete coordinates are generally not continuous variables but actually make large jumps when they change. If changes to discrete coordinates can be predicted, then both the update regions and subscription regions can take advantage of this fact to provide physically correct filtering by (1) adding a new extent early that indicates the predicted change and then (2) removing the old extent when it is no longer valid (i.e., when the discrete coordinate actually changes its value). If it is not possible to predict when discrete coordinates change their values, then the filtering will still be logically correct, but not physically correct, i.e. objects might be discovered late but it will be done in a repeatable, well-defined manner when logical time management services are used.

[8] Enumerated types are the most obvious example of this type.

Filtering on discrete coordinates can be very useful in describing categories of objects to which a federate might want to subscribe. For example, a federate might want to subscribe to objects that are radar-detectable. This could be one of the dimensions in a routing space. Radar detectability could be a dimension with two bins (True or False) which are set during initialization and never change. This kind of coordinate dimension can be useful in specifying the static characteristics of objects beyond just their class type, which is the most basic type of filtering that is provided by the declaration management services.

4.2 Filtering on Continuous Coordinates

We differentiate continuous coordinates from discrete coordinates in the sense that they can be described as continuous functions over time. In other words, they are not represented as step functions, but instead as continuous curves that smoothly change over time. We further assume that these curves can be evolved over time in software according to known equations, integrals, etc.

The RTI does not automate filtering based on known functions. It is the responsibility of the federates to change routing space regions when coordinate values in the routing space change appreciably. Future data distribution management services could automate many of the filter set-up tasks for continuous coordinates that have known functions (e.g., standard motion types, user defined functions, etc.) but this topic is beyond the scope of this discussion[9].

We assume that at any point in time, the equations can change form or characteristic parameters. However, we assume that it is possible to bound how much a routing space coordinate value can change during a specified time. In other words, if a routing space coordinate has a value v at time t, then at most it can change by δv at time $t+\delta t$. By bounding how much a continuous coordinate can vary over time, it is possible to establish efficient filtering practices that also provide physically correct filtering. To define filter specifications correctly with latency, it may also be important to bound the worst case δv for any object in the federation (which is discussed later). This has to be done for each continuous coordinate in the routing space.

The goal of this section is to discuss how much to artificially widen the subscription regions of continuous coordinate values and how often to sample the interest regions and the related update regions. It is important to recognize that in practice, one would never want to sample an object's routing space coordinates independently, but rather, their values would always be sampled together in a coordinated manner. Nonetheless, our current discussion focuses on a single routing space coordinate and how to determine its natural sampling rate. Federates need to take all of their routing space dimensions into account when determining their sampling rates. Federates may wish to further coordinate this process so that all of their local objects modify their update regions together with subscription regions.

We start this discussion by first neglecting latency and consider only the sampling of update regions and subscription regions. The key to understanding how this works is to recognize that subscription regions must be artificially expanded to always include their true region of interest within their sampling period. Similarly, update regions can (1) be widened, or (2) the subscription regions of other federates must be further expanded to account for the fact that update regions are also sampled.

---

[9] This is probably a good role for middleware.

4.2.1 Dynamically Changing Subscription Regions

First consider the sampling effects of subscription regions. Suppose that a federate's true subscription region is specified by a low and high value, $[v_{lo}, v_{hi}]$. However, when sampling the subscription region Δt time units later, the old values might have changed by as much as Δv. In order to keep the specified regions valid until the next sampling time, we must extend the subscription region by Δv on both ends[10]. In other words, the extents must be defined as $[v_{lo}-\Delta v, v_{hi}+\Delta v]$.

It is important to recognize that we could either (1) specify a value for Δv which then dictates our choice for Δt, or we could (2) specify a value for Δt which then dictates our choice for Δv.

What is the best way to define Δv and Δt?

Probably, the best way to do this is to first make sure that Δt is not too small. Otherwise federates will be required to redefine their subscription regions too frequently to be practical. Therefore, we should first define Δt_{min} as the minimum time between filter updates that makes sense. It is possible for the expanded subscription region to be much larger than the true region of interest. This is the tradeoff that one has to make concerning filter overheads and the benefits of filtering itself. Corresponding to Δt_{min} is Δv_{min}. Once Δv_{min} is determined, we might find out that Δv_{min} is still much smaller than what makes sense. For example, if routing space bins are used, and their resolutions are much larger than Δv_{min}, then without losing much efficiency (maybe none at all), we should be able to extend it further to Δv_{bin}, where Δv_{bin} is related to the bin size (probably a reasonable fraction of a bin) for that routing space dimension. Remember, the subscription regions include any cells that they might overlap in the routing space so it does not make sense to define subscription regions that are much smaller than bin sizes. In this case, the corresponding Δt should be set to Δt_{bin}.

4.2.2 Dynamically Changing Update Regions

Update regions are also sampled. It is important not to sample an update region too often in order to keep filtering efficient, but on the other hand, not too infrequently so as to make the filter ineffective. Keep in mind that it is possible for an object's routing space position to be just outside a federate's subscription region, but then as its true physical state changes, it actually moves into the federate's subscription region before the next sampling period. To account for this, either the object's position has to be represented as an extended region, or the subscription regions from other federates have to be further widened.

We have the same problem as when sampling subscription regions (i.e., how often do we sample?). Assume that the maximum amount the coordinate value can change in time δt is δv. The arguments are essentially the same as given in the subscription region sampling discussion if update regions are represented as extended regions. However, if subscription regions are expanded to account for the sampling of unextended udpate regions, then the subscription regions must be further expanded according to some known amount. This limits the choices on how to determine δv and δt. As with subscription regions, the most obvious approach is to choose δt to be not too frequent, and calculate δv based on δt. If δt is too small, then it might make sense to make it larger and extend the update regions in order to make up the difference.

---

[10] It is possible to define a different Δv for the low and high ends of a filter specification but for simplicity, we have assumed that they are the same.

4.2.3 Filtering Efficiency

The filtering efficiency of update and subscription regions can be characterized based on their extensions. For simplicity we assume a two-dimensional, geographic region with dimensions x and y. Figure 3 illustrates a more efficient subscription region and a less efficient subscription region. Even though S_1 is larger than S_2 in absolute terms, it is more efficient than S_2 because it is extended less proportionally. Conversely, if Δx_1 and Δy_1 were proportionally larger than Δx_2 and Δy_2, S_2 would be more efficient than S_1.

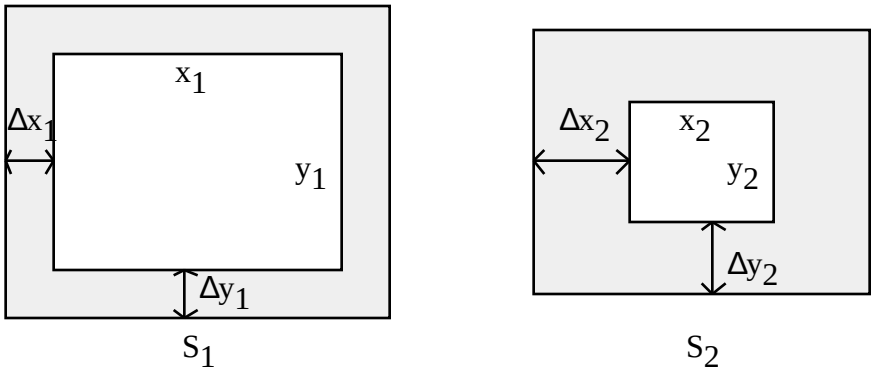

Figure 3: Characterizing the efficiency of subscription regions

Intuitively, S_1 is more efficient than S_2 because it will tend to get fewer "false positive" updates: updates which fall into the extended region, but not in the actual region. A subscription region's efficiency is:

$$\eta = xy/[(x + 2\Delta x)(y + 2\Delta y)] \quad (1)$$

The smaller the extension of the region is, the closer $2\Delta x$ and $2\Delta y$ are to zero, and the closer the region's efficiency is to one. Notice that equation (1) does not say anything about the absolute size of the subscription region.

In general, this same characterization of efficiency cannot be applied to update regions because most entities are logically single points[11]. Under such circumstances, the update region is all extension. Reflecting on our subscription region example, x and y would both be zero, resulting in the region's efficiency being zero. This is obviously not a useful measure. As we'll see in the next derivation, an update region's effect on filtering efficiency is related to the size of the subscription with which it intersects. Consider the intersection of subscription region S_1 with update region U_1 in Figure 4.

[11] Obviously this is not the case for large entities such as weather effects, but the derivations for efficiency apply equally.

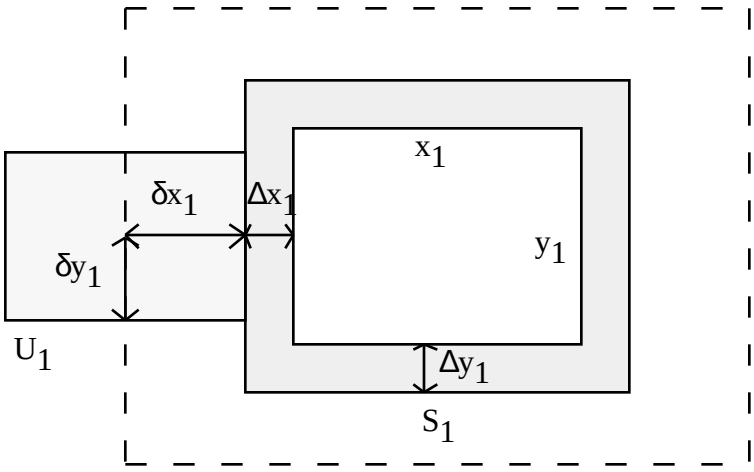

Figure 4. Filtering efficiency impact of subscription and update region intersection

The entity in U_1 is assumed to be a point in the center of the update region. If we were to extend S_1 rather than U_1 to account for the entity's rate of change and still maintain physically correct filtering, S_1's dimensions would have to be $x_1 + 2\Delta x_1 + 2\delta x_1$ by $y_1 + 2\Delta y_1 + 2\delta y_1$. The result would be yet more "false positive" updates for S_1. Plugging these extents back into our original equation for efficiency, we have that the efficiency of the filter created by the intersection of an update and subscription region is

$$\eta = xy/[(x + 2\Delta x + 2\delta x)(y + 2\Delta y + 2\delta y)] \quad (2)$$

If δx and δy are large relative to x, y, Δx and Δy the filtering efficiency will be low. If δx and δy are small relative to x, y, Δx and Δy the filtering efficiency will less impacted. So, it's not the absolute size of the update region that we use to categorize them, but their size relative to subscription regions with which they intersect. Even if an update region has a large absolute size, if all the subscription regions in the federation are several times as large, the update region's size will have little impact on filtering efficiency. Likewise, even an update region with small absolute size may have a large impact on efficiency if all the subscription regions are also small.

4.2.4 Accounting for Latencies

The discovery process takes a certain amount of time to occur. During the time L_{tot} (where L_{tot} is the total latency required for object discovery)[12], continuous coordinates may vary enough to warrant object discovery. To accommodate this fact, it is necessary for update and subscription regions to once again be artificially widened to account for the dynamic changes possible between an object's subscription region and any other object's update region. Assume that the value ε corresponds to the maximum relative change for a coordinate value v corresponding to a given latency L_{tot} (required for object discovery). Then, update and subscription regions have to be widened by the amount ε in order for them to be physically correct, i.e. the filter does not miss any objects that it should discover over time. If update regions are represented as sampled points in the routing space, the

[12] The total latency will be dependent on the algorithm used for coordinating the filtering. In real-time federates, L_{tot} involves the latencies for discovering objects. If heartbeat messages are used without push and pull handshaking for object discovery, then L_{tot} will also include the time between heartbeats.

subscription regions will have to be further widened to account for the maximum v in the federation.

4.2.5 Multiple Continuous Coordinates

We have discussed four important aspects of physically correct filtering on a continuous coordinate value. In practice, a routing space may have many continuous coordinates in its definition. Think of these continuous coordinates as a vector, $\vec{v}$. It is important to remember that updating routing space regions for these coordinate values should be done together instead of having each dimension schedule its own updates independently. Therefore, subscription regions should be sampled every Δt time units and update regions should be sampled every δt time units[13]. Because of the finite sampling rates, and due to network latencies, regions must be increased accordingly. This is shown in Table 1.

Table 1: Time values and coordinate increases.

	Time Values	Coordinate Increase
Subscription Region	Δt	$\Delta\vec{v}$
Update Region	δt	$\delta\vec{v}$
Latency	L_{tot}	$\vec{\varepsilon}$

If update regions are represented as sampled points in a routing space, the subscription regions are expanded as follows:

$$[\ \vec{v}_{lo} - \Delta\vec{v} - \delta\vec{v} - \vec{\varepsilon}\ ,\ \vec{v}_{hi} + \Delta\vec{v} + \delta\vec{v} + \vec{\varepsilon}\ ]$$

Of course, care must be taken in the above expression to ensure that the specified regions do not exceed the minimum and maximum bounds for any of the routing space dimensions[14].

5. DDM Implementations

This section is designed to provide information about specific implementations of the DDM and useful insights into their use.

5.1 Initial Implementation

The initial implementation of software to support DDM services was developed in cooperation with DARPA and the Synthetic Theater of War (STOW) Advanced Concept Technology Demonstration. This implementation will form the basis for the introduction of DDM services into the government-provided RTI software. Futher description of this

[13] There is no reason for subscription region sampling rates and update region sampling rates to be the same. In fact, it is possible for each object to define its own sampling rates based on their characteristics. For example, slow moving objects may not have to sample as often as fast movers.

[14] The choice of whether to expand update regions or just subscription regions is highly dependent on several factors including the implementation of DDM, network latencies, lookahead, and relative rates of change of update and subscription regions in the federation.

implementation is available in published papers by Van Hook, Calvin et al (see bibliography).

This implementation relies on multicast groups and TCP connections to establish communications connectivity between subscribing and publishing federates. It is important to note that it separates the process of establishing connectivity between publishers and subscribers from the process of sending data. Once the RTI finds a match between update and subscription regions and makes the connection, updates are sent directly. This separation improves performance because the RTI does not have to check every attribute against every subscription region. However, it also means that filtering is not perfect because it is at the level of regions. In the example in Figure 1, all updates in region U1 would flow to the federate that created S1, not just the updates in the portion where they overlap.

This implementation of the HLA/RTI DDM services uses a gridded approach to perform these operations. A grid is used to efficiently determine which subscription and update regions overlap as well as to efficiently calculate and setup the required network connectivity to permit data to be routed from publishers to subscribers.

Each routing space defined by a federation is partitioned into a grid of cells. The parameters of the grid for each routing space, such as cell density, are defined in a file that is read in upon initialization. All federates' RTI interfaces use the same grid parameters. The number of dimensions of each grid cell is the same as the number of dimensions in the routing space, e.g. a three dimensional routing space uses three dimensional cells.

Associated with each grid cell is a parameter called a "stream". A stream can be thought of as a logical multicast group, i.e, it defines a collection of federates that receive any data sent to the stream. A stream does not imply any particular underlying transport service such as IP multicast, TCP connections, MPI, shared memory queues, etc. It only defines groups of receivers. Streams are algorithmically assigned to cells and all federates' RTI interfaces use identical mappings of streams to cells.

Comparison of subscription regions with update regions to determine overlap for purposes of establishing connectivity is performed implicitly using the grid defined on each routing space. No explicit comparisons are performed between subscription and update region extents. Instead, each federates' RTI interface maps its regions onto a routing space grid and determines a set of streams from the cells that fall within each region. If a cell falls within a region, either partially or wholely, then that cell's stream is added to the set. Cells that lie completely outside a region are ignored.

Streams determined from mapping subscription regions to a routing space grid are joined by that federate's RTI interface. Streams determined from mapping update regions to a routing space grid are used to send attributes and interactions by that federate's RTI interface. The RTI sets up the necessary connectivity to permit data sent to a particular stream to be delivered to all federate's RTI interfaces that have joined that same stream. The implementation of this connectivity will be different depending on the actual transport service selected by the federate for each attribute or interaction class to be distributed, i.e. IP multicast groups, TCP connections, etc.

In the current design, published object-class attributes are identified with at most one routing space. Published and subscribed attributes should not be confused with the meaning of the coordinates in a routing space. Two different routing spaces, for example, can include position dimensions (along with other relevant dimensions) in their definitions. However, the two routing spaces must publish and subscribe to different attribute values. Object attributes are not allowed to span more than one routing space. This is shown in Figure 5. If an attribute were allowed to be associated with multiple routing spaces, then whenever the attribute is updated, a message for each multicast group in each routing space

would have to be sent. Subscribing hosts will receive the message several times when their subscription regions in different routing spaces overlap the update regions. All of this could be wasteful. This is why the current approach only allows an attribute to be associated with a single routing space.

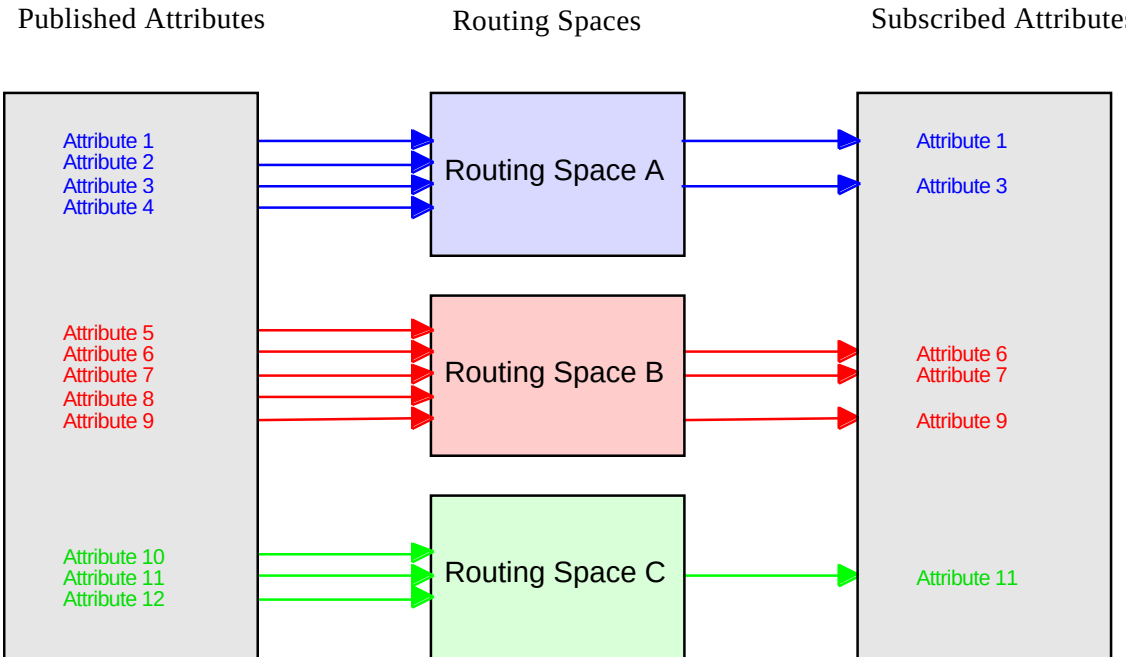

Figure 5: Mapping of object attributes to routing spaces. Publishing objects map their attributes to a routing space (an attribute cannot belong to more than one routing space). Subscribing federates do not have to subscribe to all of the published attributes in a routing space.

Each bin-cell maps to a stream which may be multiplexed with several other streams over one IP multicast group or to one or more TCP connections[15]. An object participating in the routing space normally defines its update region in one and only one bin at a time[16]. However, the subscribed interest regions of federates usually encompass several bin-cells simultaneously. When a federate's subscription region overlaps a set of bin-cells, all remote objects that are simulated by another federate and are updating in those bin-cells should be discovered by the federate. Figure 6 maps the example provided in Figure 2 to this binning approach using streams. Federate C joins the streams for the cells overlapped by the aircraft's subscription region. Updates for the ships simulated by federates A and B are sent to the streams associated with the cells that overlap each of the respective ships' update regions.

[15] The current routing space implementation uses fixed-sized bin-cells to map update and subscription regions into multicast group addresses. The choice of bin size in each dimension can strongly affect performance. The federates are able to define their bin sizes through the FED. Future data distribution management services may not be limited to fixed-sized bins.

[16] This can be extended if necessary, e.g., weather objects such as clouds could extend over several bin-cells, aggregated sets of objects could define multiple points in the routing space which span several bin-cells.

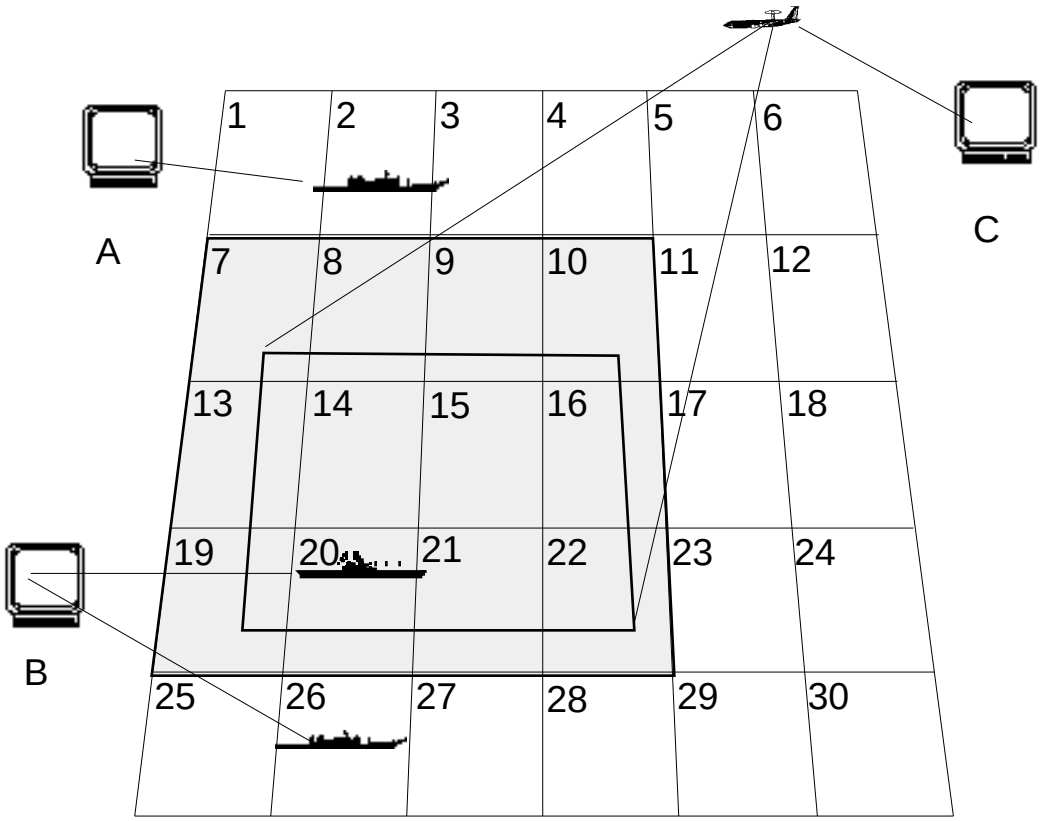

Figure 6: Mapping of update and subscription regions to streams.

The current scheme using fixed and equal sized bin-cells can be a problem when subscription regions are much larger than the bin-cell size (i.e., many bin-cells are contained within a subscription region) or much smaller than the bin-cell size (i.e., the filter will be inefficient because the resolution is too coarse). Federations should perform analysis of their filtering requirements to determine effective ranges and bin-cell sizes.

When a federate updates an object's attribute values, the RTI routes the update to the appropriate stream without knowing who the recipients are. The recipients are those federates that have subscription regions which overlap the corresponding update regions in the routing space bin-cells. All of this is transparently supported within the RTI.

6. Bibliography

James O. Calvin, Carol J. Chiang, Stephen M. McGarrry, Steve J. Rak, Daniel J. Van Hook. 1997. "Design, Implementation, and Performance of the STOW RTI Prototype (RTI-s)." In *Proceedings of the 1997 Spring Simulation Interoperability Workshop,* 97S-SIW-019.

Cisco Connection, 1996. "Documentation." Available through the Internet: *http://www.cisco.com/univ-src/3.7/home/home.htm*. Vol. 3, No. 7.

Deering S. 1989. “Host Extensions for IP Multicasting.”, *Network Working Group, rfc1112.*

DMSO 1996. “HLA Frequently Asked Questions.” Available through the Internet: *http://www.dmso.mil/projects/hla/*.

DMSO 1996. “HLA Interface Specification Version 1.0, August 15, 1996.” Available through the Internet: *http://www.dmso.mil/projects/hla/*.

DMSO 1996. “HLA Rules Version 1.0, August 15, 1996.” Available through the Internet: *http://www.dmso.mil/projects/hla/*.

DMSO 1996. “Object Model Template (OMT) Version 1.0, August 15, 1996.” Available through the Internet: *http://www.dmso.mil/projects/hla/*.

DMSO 1996. “HLA Time Management: Design Document Version 0.3, June 30, 1996.” Available through the Internet: *http://www.dmso.mil/projects/hla/*.

Fujimoto R. 1990. “Parallel Discrete Event Simulation.”, Communications of the ACM. Vol. 33, No. 10, Pages 30-53.

Mellon L. 1996. “Hierarchical Filtering in the STOW System.” In *Proceedings of the 15th DIS Workshop*.

Morse K. 1996. “Interest Management in Large Scale Distribute Simulations.” *University of California, Irvine, Technical Report TR 96-27*.

Seidensticker S. 1996. “HLA Data Filtering/Distribution Requirements.”, In *Proceedings of the 15th DIS Workshop*.

Steinman J., Wieland F. 1994. “Parallel Proximity Detection and the Distribution List Algorithm.” In *Proceeding of the 8th Workshop on Parallel and Distributed Simulation (PADS94).* Pages 3-11.

Van Hook D., et. al. 1996. “Performance of STOW RITN Application Control Techniques.”

Van Hook D., Rak S., Calvin J. 1996. “Approaches to RTI Implementation of HLA Data Distribution Management Services.”, In *Proceedings of the 15th DIS Workshop.*

Van Hook D., McGarry S. 1996. “A Prototype Approach to the Data Communications Component of the RTI.”, In *Proceedings of the 15th DIS Workshop,* (previously titled, “An Update on the RTI Design”).

Whetten B., Montgomery T., and Kaplan S. 1994. “A High Performance Totally Ordered Multicast Protocol.” *Theory and Practice in Distributed Systems*, Springer Verlag LCNS 938.

7. Terms & Definitions

API - Application Programmer’s Interface. A library of function calls which allows a federate to interact with the Run Time Infrastructure.

Associate Update Region - The mapping (or publishing) of a set of object attributes with a set of extents by a publishing federate when an object is created. This is normally done once per object during initialization.

Attribute - A named portion of an object state.

Best Effort Message Delivery - This quality of service sends messages without guaranteeing their safe delivery. Best effort message delivery services have less overhead than reliable delivery services and can therefore make better use of available network bandwidth.

Bin-Cell - Each dimension of a routing space is decomposed into a fixed number of bin-cells. This specifies the resolution of each dimension in the routing space. The total number of bin-cells in a routing space is equal to the product of the number of bin-cells for each dimension.

Broadcast - The sending of packets to all receiving hosts using a common broadcast address. Allowing the broadcast to propagate throughout the network is a significant burden on both the network (in terms of traffic volume) and the hosts connected to the network (in terms of the CPU time that each host that does not want to receive the transmission must spend processing and discarding unwanted broadcast packets). Routers can be configured to stop broadcasts at the LAN boundary (a technique that is frequently used to prevent broadcast storms), but this technique limits the receivers according to their physical location.

Continuous Coordinates - A routing space coordinate that changes its true value in a continuous manner according to some dynamic expression. An example of this might be the position of an entity (physical objects do not suddenly disappear from one location and then reappear somewhere else).

Data Distribution Management - The general term to describe mechanisms used in HLA to manage the flow of data between federates using value-based filtering mechanisms. Data distribution management is synonymous with Interest Management.

Declaration Management - The general term to describe mechanisms used in HLA to manage the flow of data between federates using the object class and attribute publication and subscription services.

Discovery - A remote object is discovered when it meets any of the interest regions specified by a federate. After discovery, data will flow through the RTI from the publisher to the subscriber.

Discrete Coordinates - A routing space coordinate that changes its true value in a non-continuous manner (i.e., large jumps are possible). An example might be an attribute that changes from an *off* state to an *on* state.

Extents - Data structure that describes a filter specification within a routing space. For each dimension of the routing space, an extent describes the begin and end value (i.e., the range) for that coordinate.

Federation Execution Data (FED) - The data required by the RTI for operation. The required data come from two distinct sources, the Federation Object Model (FOM) product, and the Federation Execution Data (FED).

Federate - Major component of an HLA federation. Federates may be simulation applications, passive viewers, control stations, data stores, etc.

Federate Queryable - Any persistent information that is stored within a federate is considered to be federate queryable because the federate can be queried for that

information. Object attributes are federate queryable but interaction events that were generated by a federate are not.

FOM - An identification of the essential classes of objects, object attributes, and object interactions that are supported by an HLA federation. In addition, optional classes of additional information may also be specified to achieve a more complete description of the federation structure and/or behavior.

Gateway - In the IP community, an older term referring to a routing device. Today, the term router is used to describe nodes that perform this function, and gateway refers to a special-purpose device that performs an application layer conversion of information from one protocol stack to another.

Glob - An update region represented in a routing space as a volume instead of as a point. Examples of globs might be: an aggregate object that is really composed of many objects distributed over space, a large object (such as an aircraft carrier), weather, or a very fast moving object (the glob always bounds the object's true position).

HLA - High Level Architecture: Defined by (1) a set of interfaces, (2) a set of rules, and (3) the Object Model Template.

IGMP - Internet Group Management Protocol: uses IP datagrams to allow IP multicast applications to join a multicast group. Membership in a multicast group is dynamic – that is, it changes over time as hosts join and leave the group. Multicast routers that run IGMP use IGMP host-query messages to keep track of the hosts that belong to multicast groups. These messages are sent to the all-systems group address 224.0.0.1. The hosts then send IGMP report messages listing the multicast groups they would like to join. When the router receives a packet addressed to a multicast group, it forwards the packet on those interfaces that have hosts that belong to that group. If you want to prevent hosts on a particular interface from participating in a multicast group, you can configure a filter on that interface by using the **ip igmp access-group** interface configuration command. Routers on which IGMP is enabled, periodically send IGMP host-query messages to refresh their knowledge of memberships present on their interfaces. If after some number of queries, the router determines that no local hosts are members of a particular multicast group on a particular interface, the router stops forwarding packets for that group and sends a "prune" message upstream toward the source of the packet. You can configure the router to be a member of a multicast group. This is useful for determining multicast reachability in a network. If a router is configured as a group member, it can, for example, respond to an IGMP echo request packet addressed to a group for which is a member.

Interest Management - The goal of interest management is to reduce the amount of information that flows between federates so that federates only receive messages that they are interested in. In HLA, the Data Distribution Management services provide the infrastructure and interfaces to support Interest Management. Interest Management is synonymous with Data Distribution Management.

Interest Region - Synonymous with Subscription Region.

IP Multicast Addressing - IP multicasting applications use Class D addresses to address packets. The high-order 4 bits of a Class D address are set to 1110, and the remaining 28 bits are set to a specific multicast group ID. Class D addresses are typically written as dotted-decimal numbers in the range of 224.0.0.0 through 239.255.255.255. Some multicast group addresses are assigned as well-known addresses by the Internet Assigned Numbers Authority (IANA). These multicast group addresses are called *permanent host groups* and are similar in concept to the well-known TCP and UDP port numbers. For example, the Network Time Protocol (NTP) uses 224.0.1.1, RIP-2 uses 224.0.0.9, and the Silicon Graphics Dogfight application uses 224.0.1.2.

LAN - A Local Area Network is a high-speed, low-error data network covering a relatively small geographic area (up to a few thousand meters). LANs connect workstations, peripherals, terminals, and other devices in a single building or other geographically limited area. LAN standards specify cabling and signaling at the physical and data link layers of the OSI model. Ethernet, FDDI, and Token Ring are widely used LAN technologies.

Latency - The delay between when a message is sent from the source to when it is received by the destination.

Message - A data unit transmitted between federates containing at most one event. Here, a message typically contains information concerning an event, and is used to notify another federate that the event has occurred. When containing such event information, the message's time stamp is defined as the time stamp of the event to which it corresponds. Here, a "message" corresponds to a single event, however the physical transport media may include several such messages in a single "physical message" that is transmitted through the network.

Minimum Rate Message Delivery - The minimum rate transport service most closely emulates DIS transport mechanisms. Attributes are sent at a minimum rate even if no attribute values have changed.

Modify Region Service - This service allows a federate to dynamically modify its update and subscription regions in a routing space.

Multicast - The sending of each message packet to a multicast group address. Hosts that want to receive the packets indicate that they want to be members of the multicast group. Applications sending multicast packets expect that networks with hosts that have joined a multicast group will receive those multicast packets. Multicast applications and underlying multicast protocols often control multimedia traffic and shield hosts from having to process unnecessary broadcast traffic. In large military simulations involving large numbers of hosts, multicasting is used to transmit information efficiently between federates according to the interest management strategy that has been set up.

Object - A fundamental element of a conceptual representation for a federate that reflects the "real world" at levels of abstraction and resolution appropriate for federate interoperability. For any given value of time, the state of an object is defined as the enumeration of all its attribute values.

Object Management - The group of RTI services which deal with the creation, modification, and deletion of objects and the interactions they produce.

Physical Correctness - Physically correct filters ensure that all objects that should be discovered, are in fact discovered. Physically correct filters may not always be efficient (i.e., a federate may discover many objects that it really is not interested in), but they never miss any of the objects that they are truly interested in.

Proximity Detection - A subset of the more general interest management problem that focuses only on objects detecting each other based on their relative distance. Proximity detection is usually the most important filtering mechanism needed in distributed military simulations. Proximity detection is sometimes called Range Based Filtering.

Publish - In HLA, object classes that are defined in the FOM contain attributes that are publishable. Those attributes that are published by one federate then become available to other federates through the subscription process.

Push/Pull Methods - Push methods "push" their information across the network to other federates. Pull methods request information from other federates. Note that push methods may send extra information but require only one hop. Pull methods require two hops (the request and the response).

Range Based Filtering - See Proximity Detection.

Real Time Simulations - A simulation where time advances are paced to have a specific relationship to wall clock time. These are commonly referred to as real-time or scaled-real-time simulations. Here, the terms *constrained simulation* and *(scaled) real-time simulation* are used synonymously. Human-in-the-loop (e.g., training exercises) and hardware-in-the-loop (e.g., test and evaluation simulations) are examples of constrained simulations

Region - A data structure defined during run time to represent a subset of a routing space. It includes the name of the routing space from which it is derived and an extent record that describes each of the dimensions. Regions are specified using a component of the Data Distribution Management services.

Register Object - Objects are created and managed within each federate. The official way for the RTI to know about the creation of such objects is for the federate to invoke the Register Object service from the RTI for each of its objects that are discoverable by any of the other federates in the federation. The kinds of objects that can be registered, along with their attributes, must be described in the FOM.

Reliable Message Delivery - Message delivery services which ensure that all messages are correctly received by each of the subscribing federates. Reliable message delivery services are generally more expensive than best effort services but are usually required for logically correct federates.

Router - Network layer device that uses one or more metrics to determine the optimal path along which network traffic should be forwarded. Routers forward packets from one network to another based on network layer information. Occasionally called a gateway (although this definition of gateway is becoming increasingly outdated).

Routing Space - An abstract multidimensional coordinate system that is used to represent dynamically changing update and subscription interest regions (see Update Region and Subscription Regions).

Routing Space Coordinate - Routing spaces are composed of multiple routing space coordinates (or dimensions). Each coordinate represents a different kind of state variable used for filtering. For example, (X, Y, Radar Cross Section) might represent three coordinates in a routing space. These coordinates do not have to be related to the attributes that are published.

Routing Space Dimension - Same as Routing Space Coordinate.

Run Time Infrastructure (RTI) - The general purpose distributed operating system software which provides the common interface services during the runtime of an HLA federation.

Sampled Values - Each federate is required to pass sampled routing space coordinate values to the RTI in order to describe update regions and subscription regions because the RTI does not know how to evolve these quantities itself. It is important to sample at a rate that makes sense. If values are sampled too often, the overhead in setting up extents, etc., will make filtering ineffective. If values are sampled too infrequently, then extents may have to be widened to unacceptable levels, thereby making filtering inefficient.

Scalability - The measurable effectiveness of a distributed system's ability to easily cope with the addition of users and sites, and to grow with minimal expense, performance degradation, and administrative complexity.

Send Interaction Service - This service provides a general event scheduling mechanism between federates. In HLA, interactions are thought of as short-lived objects that exist at a point in time and then disappear. Interaction events are different from update attribute events in the sense that they don't reflect state variables that have changed during

the course of local event processing in a federate, but instead, they are used to schedule a future interaction/computation to be performed in another federate.

Space Handle - A handle (e.g., an integer) that identifies a routing space. A Space Handle maps in a one-to-one manner to a *Space Name* (the RTI provides this mapping through the *getSpaceHandle* service).

Space Name - The string name of a routing space. A Space Name maps in a one-to-one manner to a *Space Handle* (the RTI provides this mapping through the *getSpaceName* service).

State Consistent Message Delivery - The state consistent transport service ensures delivery of the latest attribute values. It does not guarantee delivery of any intermediate attribute updates, however. The state consistent service is appropriate for attribute values that change infrequently.

Subscribe - Federates subscribe to information that other federates publish. The simple object class attribute-based subscription is usually done at a high level by the federate. The more elaborate routing space subscription mechanisms are normally done on a per-object basis where each object specifies its own interest regions. It is possible for a federate to manage its subscriptions in an aggregated manner (e.g., if it knows that all of its objects move together in a group, etc.).

Subscription Region - A region in a routing space that (normally) describes the interests of a subscribing object. A subscribing object may require several subscription regions to describe its interests, possibly in several routing spaces. It is also possible for a federate with global knowledge of its objects to describe interest regions for a group of its internal objects.

Thresholds - Relates how far routing space coordinates can change before requiring an update to their regions. Usually, thresholds are used to provide information to a federate concerning how extent values relate to bin-cell boundaries.

Transportation Service - An RTI provided service for transmitting messages between federates. Different categories of service are defined with different characteristics regarding reliability of delivery and message ordering, including best effort, causal, state consistent, and reliable.

Unfair Fight - Any situation where one federate gains an unfair advantage over another federate due to circumstances that have nothing to do with the physical system that is being modeled. Examples of an unfair fight might be: one host is faster than another host so it makes its decisions quicker and therefore has an advantage over slower hosts; one host gets its messages earlier than another host which allows its federate to gain access to information earlier than another federate; one federate has smaller lookahead value which allows it to interact more tightly in time (for example, shoot a weapon) than another federate.

Unicast - Sending of one copy of each packet to each host that needs to receive the packet. This type of communication is easy to implement, but it requires extra bandwidth because the network has to carry the same packet multiple times – even on shared links.

Update Attributes - State variables or attributes that belong to simulated entities are generally made public to other federates as published attributes that are defined in the FOM. When state variables are modified by a federate due to its internal event processing, the other federates need to be informed of these changes. To accomplish this, the sending federate invokes the update attribute service provided by the RTI. The receiving federates will have their reflect object attributes service invoked.

Update Region - The location of an object in a routing space as a function of time. Update Regions represent the publishing object's interest in the routing space. Subscribers will discover the object if their subscription regions overlap the publishing object's update region. Update regions are sampled by the federate and are then used to redefine the interests of the object in the routing space. The RTI does not automatically move objects through routing space bin-cells based on dynamic expressions because it does not know the meaning of federate related variables.

Value Based Filtering - Filtering based on values that can change during the course of a simulation. The Lincoln Labs Routing space approach is one example of a value based filtering strategy.

Wall Clock Time - A federate's measurement of true global time, where the measurement is typically output from a hardware clock. The error in this measurement can be expressed as an algebraic residual between the wall clock time and true global time or as an amount of estimation uncertainty associated with the wall clock time measurement software and the hardware clock errors.

WAN - Wide Area Network: a general term used to describe any form of network - private or public - that covers a broad geographical area.